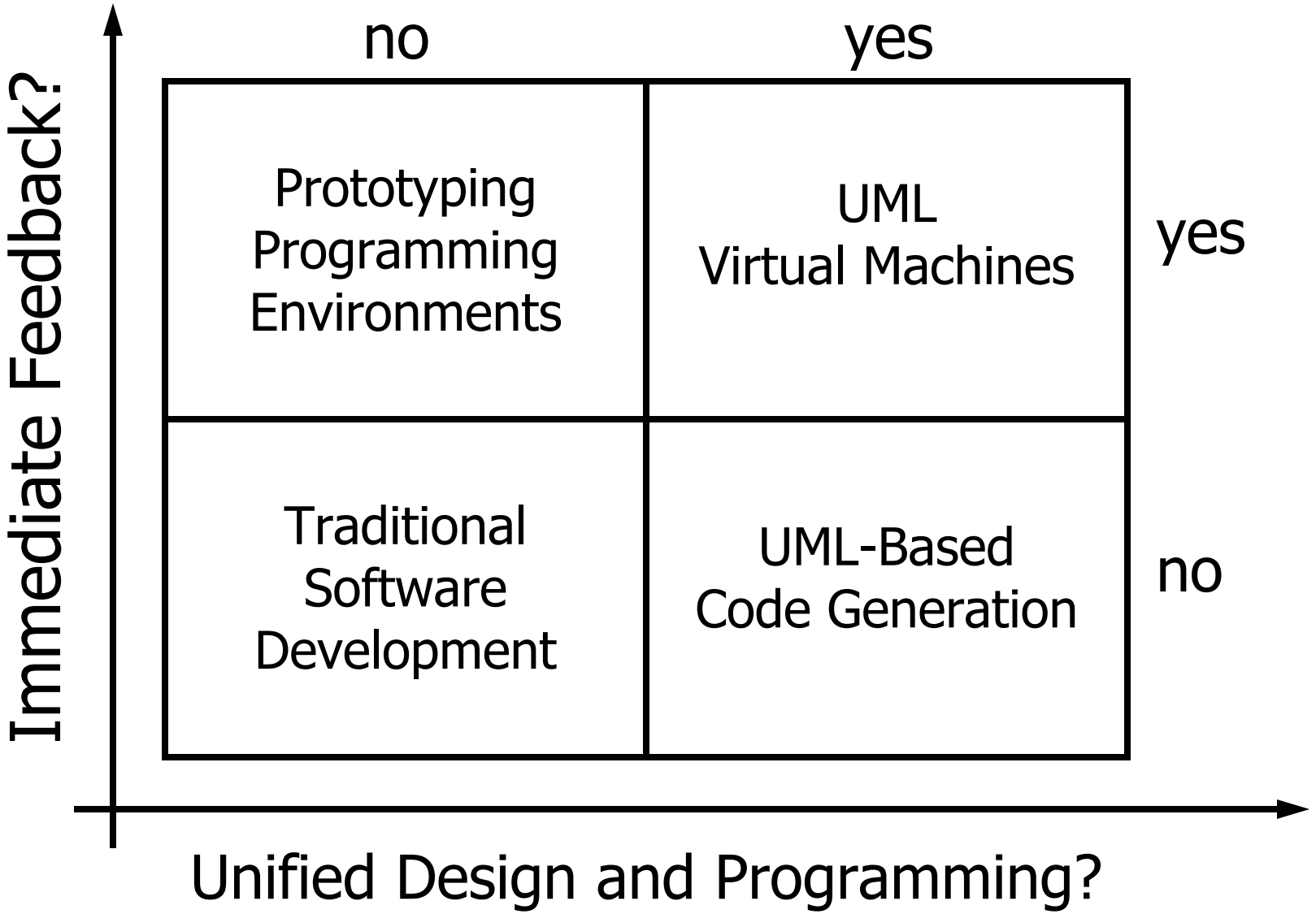
The Architecture of a UML Virtual Machine

Dirk Riehle
dirk@riehle.org, www.riehle.org

Presented to Java User Group Switzerland, December 2003
Last updated .

Motivation for UML Virtual Machines



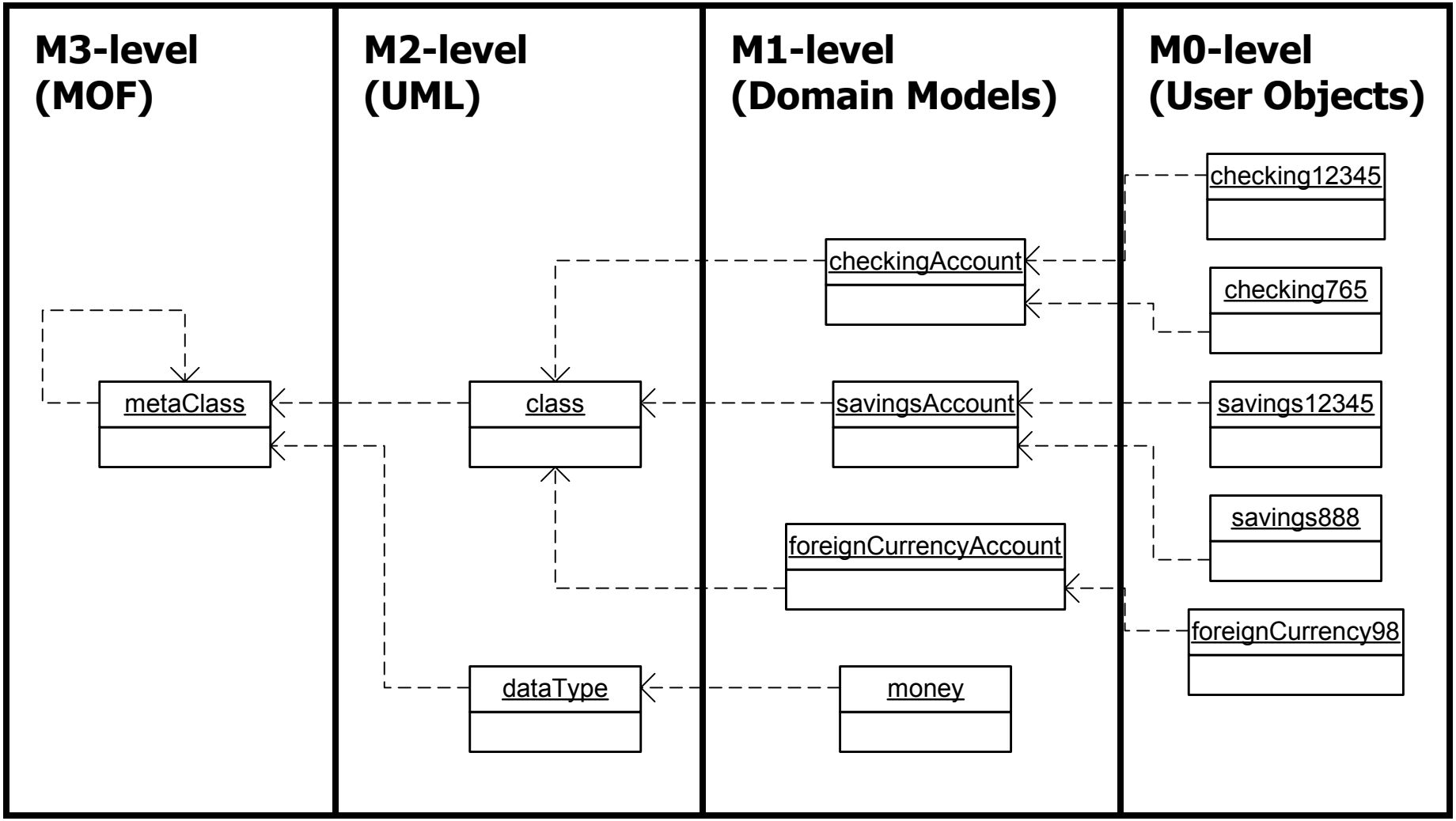
Definition: UML Virtual Machine

- UML virtual machine
  - Is an abstract computing machine (like any VM)
  - Provides an instruction set and a memory model for representing objects
- Instruction set of a UML virtual machine
  - Behavior modeled using UML itself, complemented by Java
  - Elements are persistently represented using XMI
- Memory model of the virtual machine
  - Memory management facilities of implementation language (Java)
  - Dedicated packages, garbage collection

High-level Requirements for UML VM

- Model representation
  - Represents models on all levels
- Model execution
  - Interprets models for execution
- Causal connection between models
  - Changes have immediate and defined effects

UML 4-Layer Architecture (Logical Arch.)

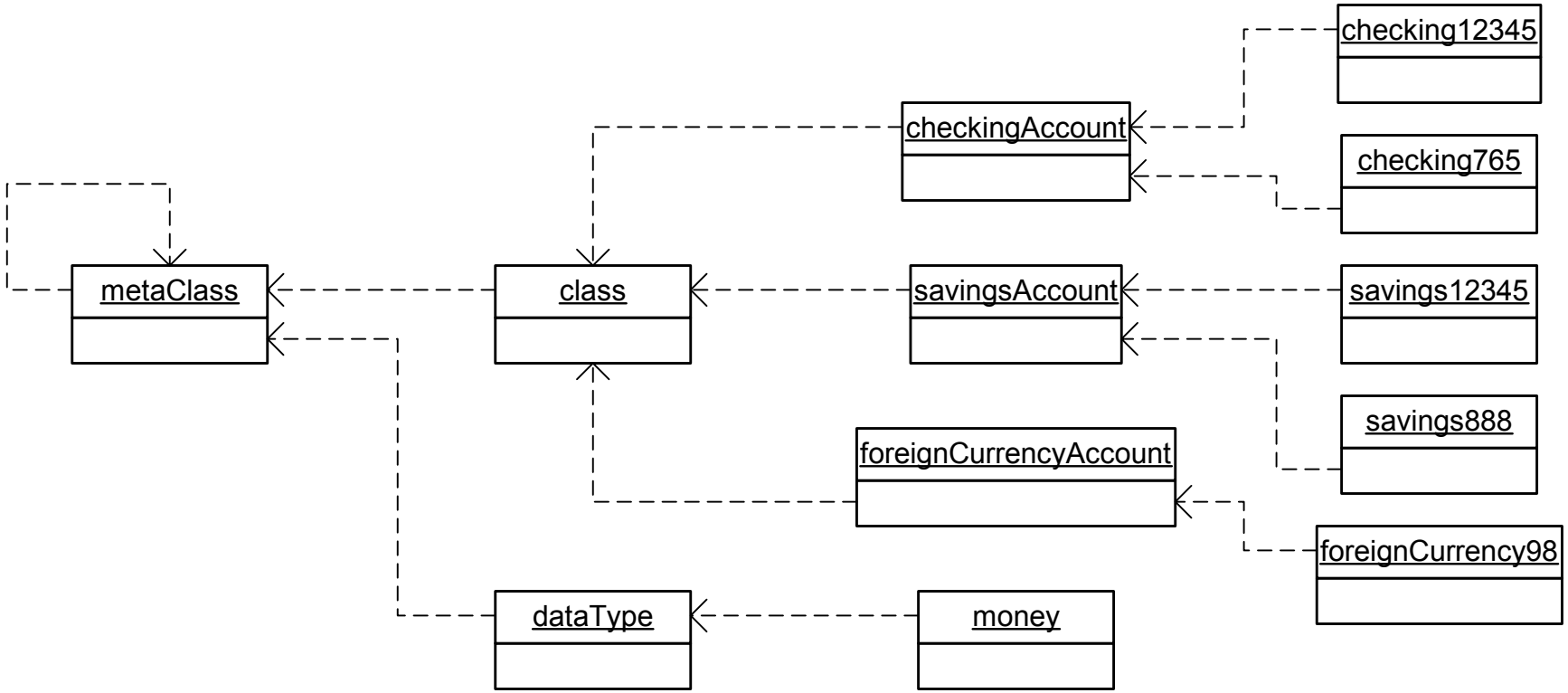


Causal Connection

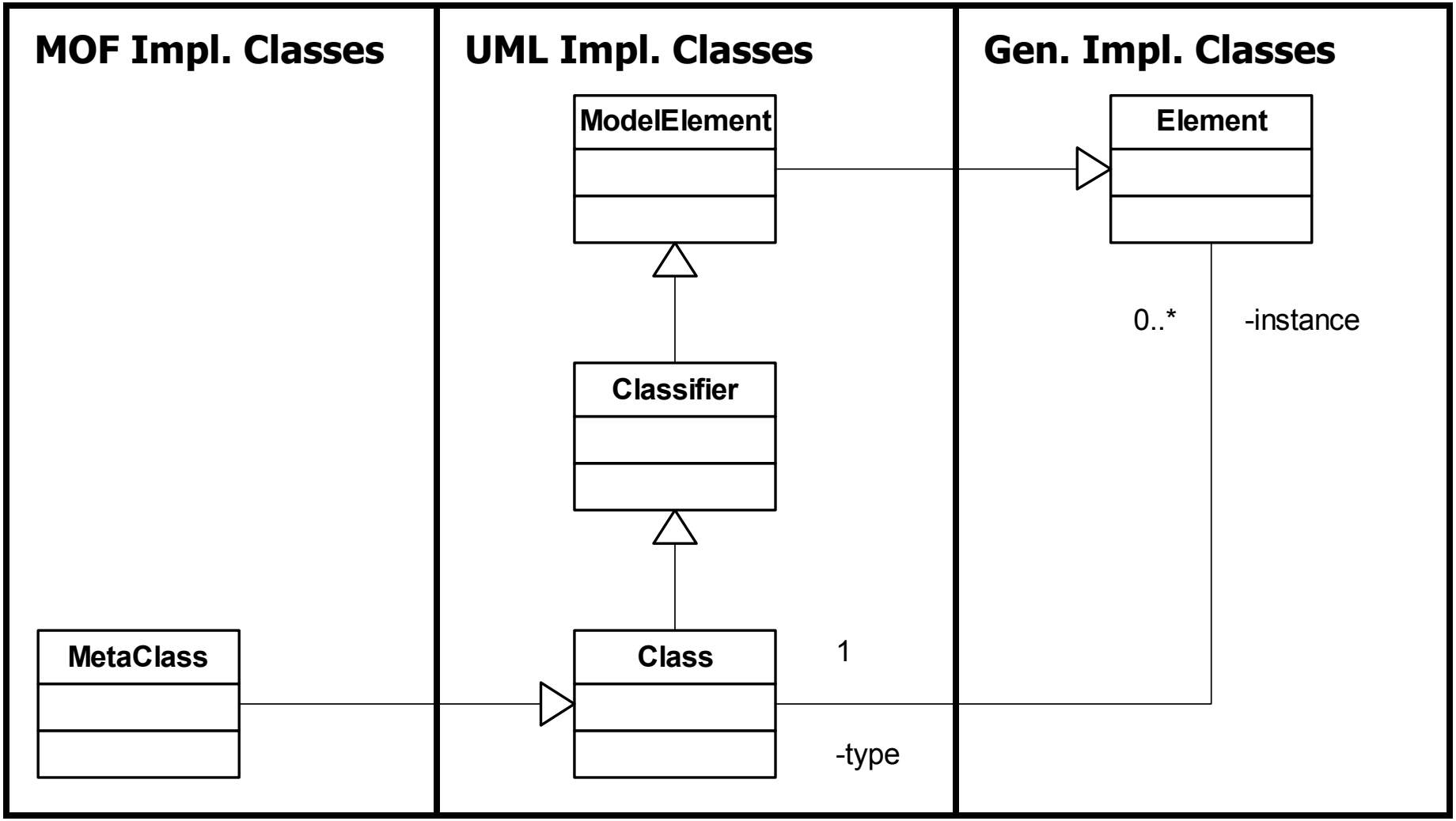
- Definition: Causal Connection
  - A modeling level is causally connected with the next higher modeling level, if the lower level conforms to the higher level and if changes in the higher level lead to according changes in the lower level
- Consequences for UVM
  - Immediate feedback to model changes

Single Causally Connected Model

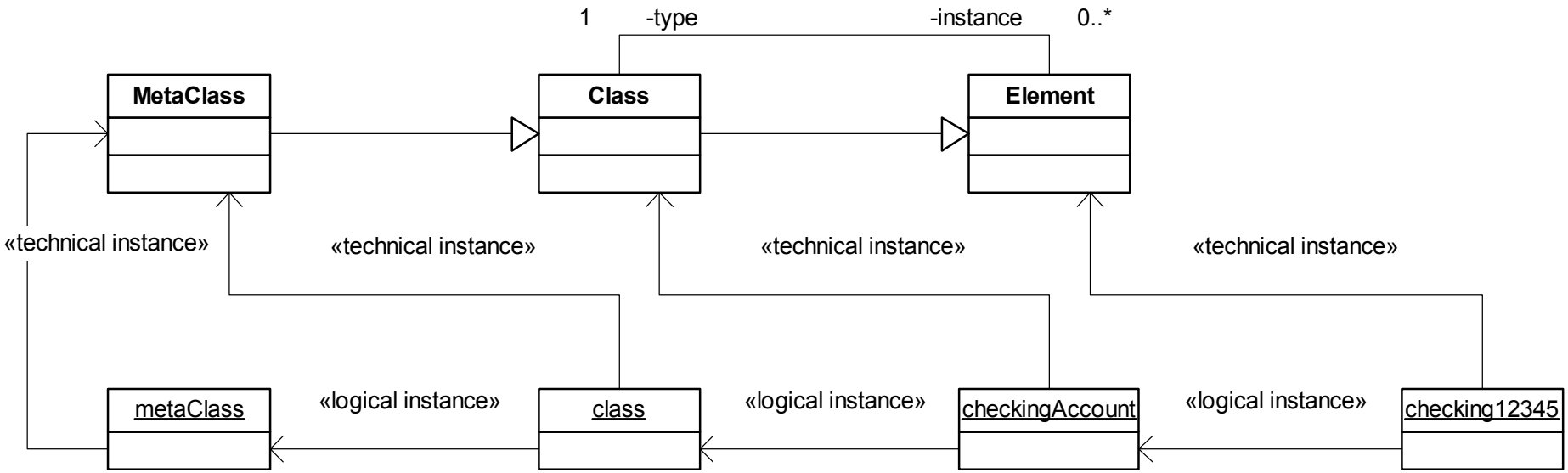
Modeling and Execution Environment



Java Implementation (Physical Arch.)



Mapping Logical/Physical Architecture



Ensuring Causal Connection

- Every element has a reference to its class
- Every access to an element is type-checked by its class

=> Every M0-element is causally connected with its M1-class

- Every class is an element

=> Every Mn-element is causally connected with its Mn+1 class

Element Functionality

- UML-derived functionality
  - Generic attribute value and link access
  - Type checking that can be switched on or off
  - Handling of association objects
  - …
- VM-needed functionality
  - Shallow/deep cloning, equality
  - Backpointers, garbage collection
  - Serialization, inspection
  - Team collaboration state model, versioning
  - External resource management
  - …

Behavior Modeling and Execution

- UML state machines
  - State machines are most precisely defined
  - Best understood how to implement
- OCL enhancements
  - Ensure constraints and business rules
  - Lightweight "programming" tasks
- Hand-programming (Java)
  - Customization through policies
  - Requires well-designed extension architecture

SKYVA Runtime Environment

- Predefined models with predefined implementations
  - Technical as well as business models (some proprietary, some not)
  - Includes configuration models
- Is paramount for efficient execution
  - Transactions, persistence, security!
  - Today: San Francisco; Soon: IBM Websphere

Project Experiences

- Use of UML virtual machine
  - Low performance tasks only (e.g. system configuration)
  - Underlies merged modeling and execution environment
- Use of code-generation approach
  - For high-performance tasks (i.e., production system)
  - Fully separated environments; different evolution strategies

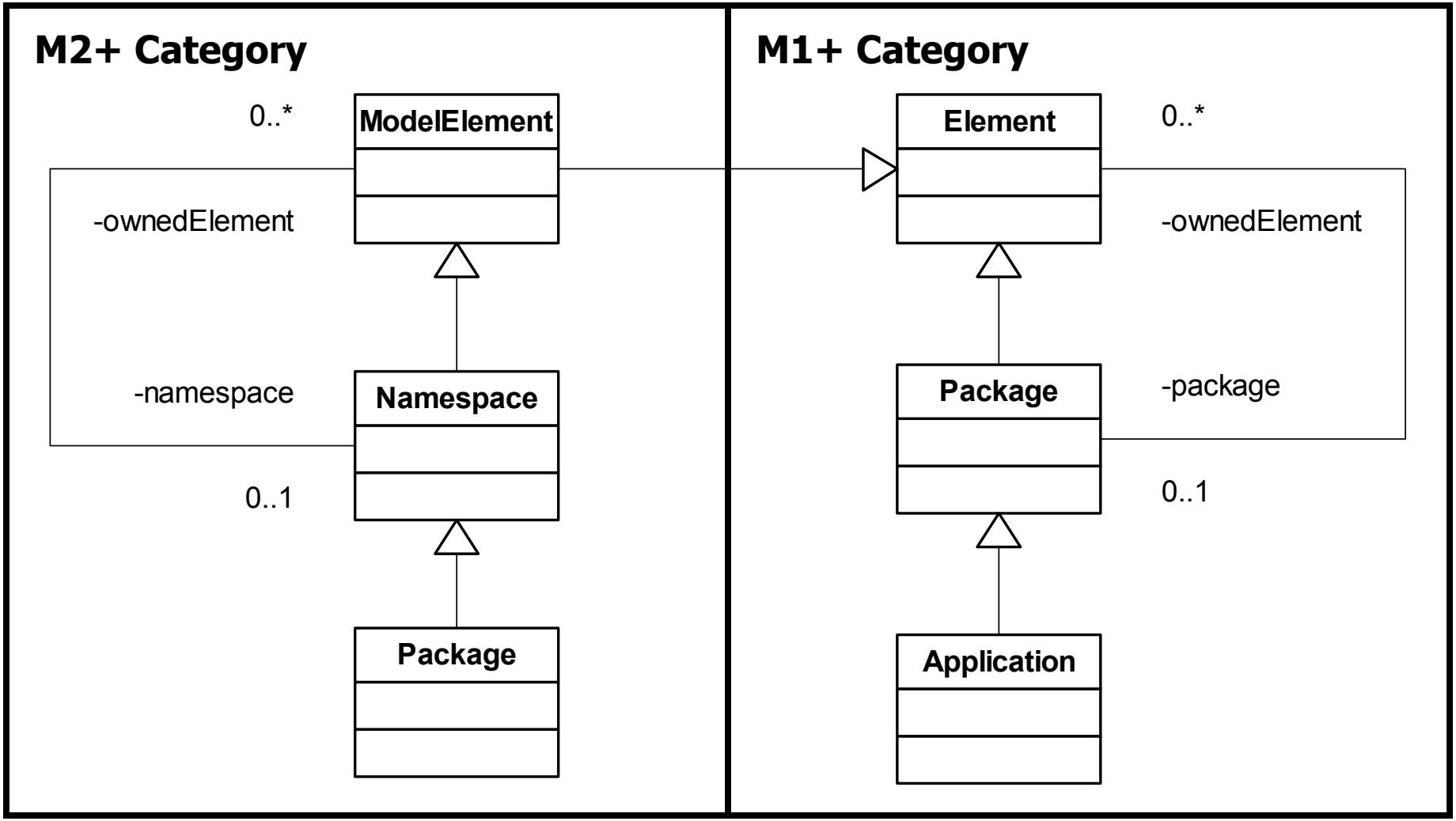
Implementation

- Using keys rather than full-fledged runtime type-checking
  - Keys only change if type information changes
- Keys speed up lookup
  - Perfect hash function (lookup in array)
- Keys speed up type checking
  - Keys carry type information

UML Simplifications

- No multi-valued attributes of elements
- Attributes of elements are always UML data type instances
- Only single inheritance between classes
- Only binary associations between classes

UML Enhancement: Application



Open Issues

- Improve behavior modeling capabilities
- Integrate Action Semantics as Java replacement
- Improve execution speed

Bare-Bones UML Virtual Machines?

- Bare-bones UML Virtual Machine
  - Knows only UML as basis for domain models
  - Can only execute behavior of standard types of classes
  - Does not allow for domain-specific UML extensions
- Advantages
  - Easier to implement, better performance
- Disadvantages
  - Forbids use of profiles in modeling

Missing Pieces 1/2

- Element class
  - Element/Class collaboration specification
- Behavioral specification of UML itself
  - Time-honored tradition: eating your own dog food
- Operational model of VM
  - Starting point (main)
  - Life-cycle model (incl. garbage collection)
  - Concurrency model
  - Model evolution support
  - Native call interface

Missing Pieces 2/2

- Standardized technical models (libraries)
  - GUI library
  - Simple persistence

Summary

- Expect UML Virtual Machines
- Expect UML IDEs/systems like Smalltalk

- See www.riehle.org/papers/2001/oopsla-2001.html
- Also see: yahoogroups.com/group/uml-virtual-machines
- If you have questions, feel free to email: dirk@riehle.org

- If you (intend to) develop UML VMs…

- Personal note: check out www.jvalue.org